Email SPAM filtering using Machine Learning

Vishal Khatri, Manas Tripathi
vishalak@uw.edu, manas93@uw.ed
University of Washington, Information School

Abstract— **Email spam or junk e-mail (unsolicited bulk e-mail) is one of the major problems of the today's Internet and is a threat to systems, organizations and people. The increasing volume of SPAM over the years has generated a need for reliable SPAM filtering techniques. Machine learning is one of the popular methods used to achieve SPAM filtering. In this paper, we evaluate the performance of various machine learning models for SPAM filtering.**

Keywords—SPAM, SPAM Filtering, Naive Bayes, Logistic Regression, k-Nearest Neighbors, Accuracy, AUC, ROC

I. Introduction

Electronic-mail (abbreviated as e-mail) is a fast, effective and inexpensive method of exchanging messages over the Internet. Despite the rise of instant messaging technologies, email continues to be a dominant mode of communication for both consumer and business use. According to the Internet Security Threat Report (ISTR) (Symantec, 2018) published by Symantec Corporation in 2018, more than 200 billion emails were sent each day in 2017. With the rapid increase in email usage, there has also been increase in the SPAM emails. Email SPAM also known as junk email, is unsolicited messages sent in bulk (Wikipedia, 2019). These SPAM emails pose a security threat to systems, individuals and organizations. According to ISTR, 55% of the emails sent in 2017 were SPAM. On an organizational front, spam effects include: i) annoyance to individual users, ii) less reliable e-mails, iii) loss of work productivity, iv) misuse of network bandwidth, v) wastage of file server storage space and computational power, vi) spread of viruses, worms, and Trojan horses, and vii) financial losses through phishing, Denial of Service (DoS), directory harvesting attacks, etc. With an increase in the number of SPAM emails being sent out every year, we can say that the need for reliable anti-SPAM filters remains high.

Two common approaches popular within academia and researchers for building anti-SPAM filters are 'Knowledge Engineering' and 'Machine Learning'. In knowledge engineering approach a set of rules must be specified according to which emails are categorized as spam or ham. These rules should be created either by the user of the filter, or by a third-party software vendor. The primary disadvantage of this method is that these rules need to be maintained and updated continuously which is a waste of time and decreases the effectiveness of the filters. The Machine Learning approach is found to be better because it doesn't require any specific rules but instead a set of emails that have been successfully classified as SPAM and NOT-SPAM (Ham). These emails are then used as a training set for a machine learning algorithm to build a model. This model then uses the information gained from the training set to classify future emails as SPAM/Ham. There are several machine learning algorithms that can be used for email-filtering like Logistic Regression, Naïve-Bayes, Support Vector Machines, Neural Networks, K-Nearest Neighbors etc. In order to build our in-house SPAM filter, we have evaluated the Naïve Bayes, Logistic Regression and K-Nearest Neighbors algorithms.

.

II. Methods

A. Data Collection

Collecting the training data for our machine learning algorithms was the first step. The dataset was retrieved using the Google's Gmail API and it consists of

- 4000 emails from inbox that are not classified as SPAM. The headers and metadata from the emails were removed and the email body was obtained in plain text. These email texts were stored with a label 'Ham'.
- 4000 SPAM emails from the same user account. Images, hyperlinks and metadata information were removed, and email body was stored as plain text against the label 'Spam'.
- The number of 'Ham' and 'Spam' messages were kept the same in order to ensure that the resulting machine learning model is not skewed.

B. Data Cleaning and Preparation

One of the major steps in Data Analytics is data cleaning. Cleaning (or in other words preprocessing) is a method to ensure a certain level of quality of data that can be used in order to make precise and accurate predictions from a statistical model (Loo & Jonge, 2018). Below are the steps we performed for data cleaning and preparation

1. **Merging the retrieved emails -** The data was collected in 2 different files (One for Spam emails and one for Non-Spam emails). We started by merging the two sets into one.
2. **Lower-casing -** In order to ensure that our model performance is not degraded because of same words being in different cases, we changed all our email text data to lowercase.
3. **Removing Punctuations and Stopwords -** To clean the lowercase data, we removed punctuations and stopwords using NLTK corpus.
4. **Vectorization -** In this step, we used the TfidVectorizer to collect words and their corresponding frequencies in the emails to create a sparse matrix.

5. **Preparing the training and test datasets -** We shuffled the available dataset to get a truly random dataset and split it into 2 parts. The first set comprised of 67% of the data and was used as the training set for building machine learning models. The second set comprised of the remaining 33% of the data and was used to test and evaluate the performance of the models built using the training set.

C. Machine Learning Algorithms

For building our SPAM filter, we evaluated 3 different machine learning algorithms - Naive Bayes, Logistic Regression and k-Nearest Neighbors. These algorithms are discussed below

Naive Bayes:

Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. It is a supervised machine learning algorithm based on applying Bayes theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. This algorithm works well for text classification problems and hence is a good candidate for creating our Spam filter. The algorithm determines the probability of the features occurring in each class and returns the most likely class. This is done using the Bayes theorem as follows (scikit-learn, 2019)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

If we have a class variable y and dependent feature vector x_1 through x_n, we get

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

If we use the naïve conditional independence assumption, the equation reduces to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

The next step is to prepare the model. We run it for each class and pick the output with maximum probability

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Advantages of Naive Bayes in SPAM Filtering:

1) It is easy and fast to predict the class of the test data set.

2) It can be trained on a per-user basis. Since the Spam a user receives is often related to the online user's activity, a Bayesian spam filter will eventually assign a higher probability based on the user's specific patterns.

3) When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and we need less training data.

Disadvantages of Naive Bayes in SPAM Filtering

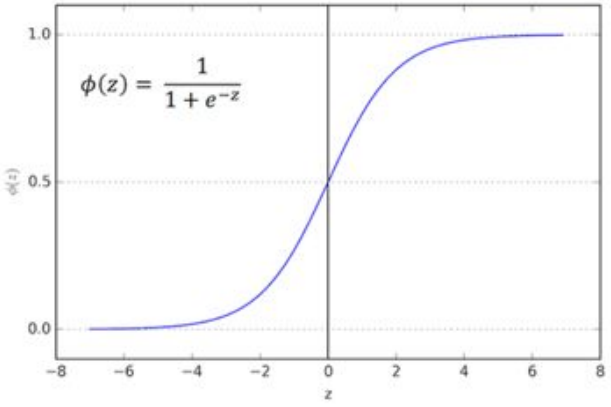
1) Spammers can get around Bayesian spam filters by replacing text with images. The whole text of the message, or some part of it, is replaced with a picture where the same text is "drawn".

2) If there is a category in the test dataset which was not observed in the training dataset, the model will assign a zero probability and will be unable to make a prediction. This is often known as Zero frequency.

3) Spammers might transform spam words in order to reduce their frequencies. For example, Lottery can be spelled as Looootterrryyy or Lotte₹₹y. The recipient of the message can still read the changed words, but each of these words is met more rarely by the Bayesian filter, which hinders its learning process.

Logistic Regression:

Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function. It can be useful in spam detection because we need to predict whether the email is spam (0) or not (1). The algorithm is based on the logistic function which is shown below with the Sigmoid curve



The logistic function gives an S-shaped curve as shown above. The output of logistic function tends towards 1 as z → ∞ and tends towards 0 as z → −∞. Hence the logistic function produces the value of dependent variable which will always lie between [0,1] i.e. the probability of being in a class. (Wikipedia, 2019)

Advantages of Logistic Regression in SPAM Filtering:

1) Outputs have a good probabilistic interpretation and the algorithm can be regularized to avoid overfitting.

2) Logistic models can be updated easily with new data using techniques like stochastic gradient descent.

Disadvantages of Logistic Regression in SPAM Filtering

1) Logistic Regression tends to underperform when there are non-linear decision boundaries.

2) It is not flexible to adapt to more complex relationships.

k-Nearest Neighbors (KNN):

The KNN classifier is considered an example-based classifier, that means that the training documents are used for comparison rather than an explicit category representation. As such, there is no real training phase. When a new document needs to be categorized, the k most similar documents (neighbors) are found and if a large enough proportion of them have been assigned to a certain category, the new document is also assigned to this category, otherwise not. To decide whether a message is spam or ham, we look at the class of the messages that are closest to it. The comparison between the vectors is a real time process. (James, Witten, Hastie, & Tibshirani, 2013)

Advantages of KNN in SPAM Filtering:

1) KNN can model complex decision boundaries

2) Since KNN is a lazy-learning algorithm, the training time is negligible.

Disadvantages of KNN in SPAM Filtering

1) Since prediction for a new observation is made by comparing with every instance in the training set, the process is memory intensive and time consuming.

2) KNN is outlier sensitive

D. Implementation and Comparison of Algorithms

In order to compare the three algorithms, we built a model using each of the algorithms and our training data. Then we ran the same model against our testing data to determine the accuracy and to build the confusion matrix.

Below are the snippets showing the implementation for each individual model

Naïve Bayes:

```
#building the Multinomial Naive Bayes model
string = ''
vectorizer = TfidfVectorizer()
vectorize_text = vectorizer.fit_transform(X_train)
MNBmodel = MultinomialNB()
MNBmodel.fit(vectorize_text, y_train)

# score
vectorize_text = vectorizer.transform(X_test)
score = MNBmodel.score(vectorize_text, y_test)
result_dict['Multinomial Naive Bayes'] = [score]
string += 'The Multinomial Naive Bayes Model has mean accuracy: ' + str(score)
print(string)

y_pred = MNBmodel.predict(vectorize_text)
print('Confusion Matrix below : ')
metrics.confusion_matrix(y_test, y_pred)
```

```
The Multinomial Naive Bayes Model has mean accuracy: 0.8299242424242425
Confusion Matrix below :

array([[1161,  183],
       [ 266, 1030]], dtype=int64)
```

Logistic Regression:

```
# building the logistics regression model
string = ''
vectorizer = TfidfVectorizer()
vectorize_text = vectorizer.fit_transform(X_train)
LRmodel = LogisticRegression()
LRmodel.fit(vectorize_text, y_train)

# score
vectorize_text = vectorizer.transform(X_test)
score = LRmodel.score(vectorize_text, y_test)
result_dict['Logistic Regression'] = [score]
string += 'The Logistic Regression Model has mean accuracy: ' + str(score)
print(string + '\n')

y_pred = LRmodel.predict(vectorize_text)
print('Confusion Matrix below : ')
metrics.confusion_matrix(y_test, y_pred)
```

```
The Logistic Regression Model has mean accuracy: 0.8431818181818181

Confusion Matrix below :

array([[1074,  270],
       [ 144, 1152]], dtype=int64)
```

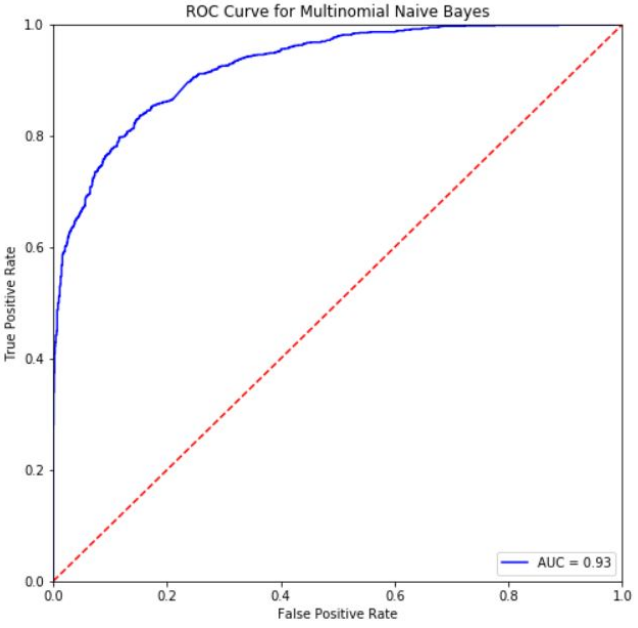
KNN:

```
#building the k-nearest neighbors model
string = ''
vectorizer = TfidfVectorizer()
vectorize_text = vectorizer.fit_transform(X_train)
KNmodel = KNeighborsClassifier(n_neighbors=5)
KNmodel.fit(vectorize_text, y_train)

# score
vectorize_text = vectorizer.transform(X_test)
score = KNmodel.score(vectorize_text, y_test)
result_dict['K-Nearest'] = [score]
string += 'The k-nearest neighbors model has mean accuracy: ' + str(score)
print(string)

y_pred = KNmodel.predict(vectorize_text)
print('Confusion Matrix below : ')
metrics.confusion_matrix(y_test, y_pred)
```

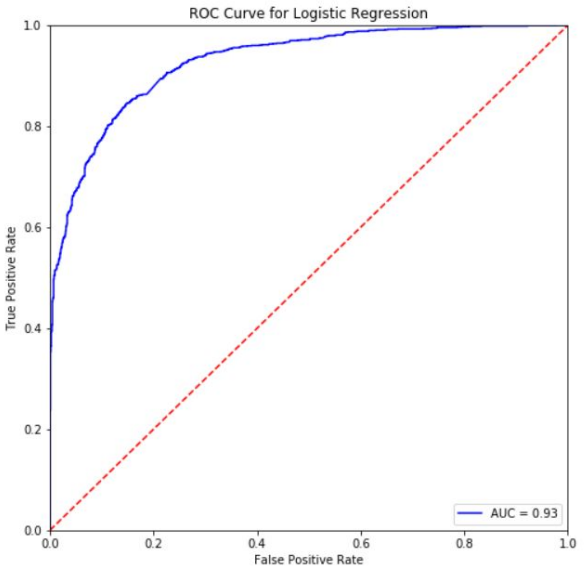
```
The k-nearest neighbors model has mean accuracy: 0.6537878787878788
Confusion Matrix below :

array([[1006,  338],
       [ 576,  720]], dtype=int64)
```

To further evaluate the performance of our models, we plotted the ROC and AUC curves

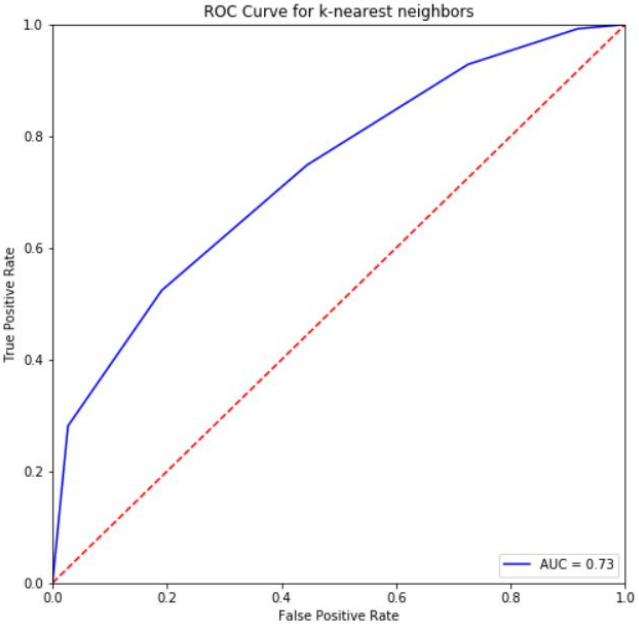
Naïve Bayes:



Logistic Regression:



KNN:



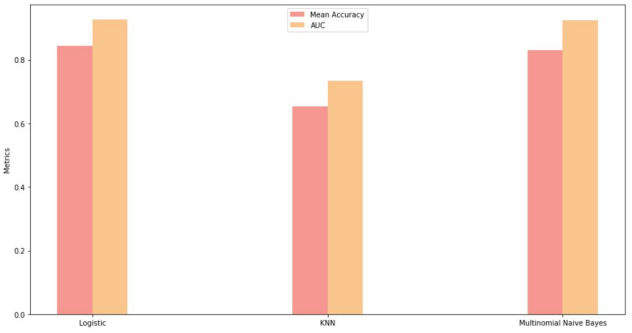
III. Results

The mean accuracy of each classifier model is shown below.

Classifier	Mean Accuracy
Multinomial Naive Bayes	0.8299
Logistic Regression	0.8431
KNN	0.6537

Classifier	AUC
Multinomial Naive Bayes	0.9317
Logistic Regression	0.9322
KNN	0.7230

From the plot in Fig 2., Below is the plot for the model metrics for each classifier.



.

From the above graph, we can see that KNN doesn't perform well in the case of SPAM filtering as the other two algorithms. The ROC plot for NB and Logistic Regression lie above the ROC plot for KNN (they **do not cross** the ROC plot for KNN), therefore they perform better than KNN as all possible FPR values we have a high TPR.

For Naive Bayes and Logistic regression, we see that the accuracy and AUC values are really close with Logistic Regression having an ~1.5% higher accuracy. It is known that when you have very little data a generative model such as Naive Bayes performs better than a discriminative model such as Logistic Regression. (Ng & Jordan, n.d.)

In the particular case of spam detection, the number of emails can be huge and therefore we might want to consider Logistic Regression model for our classification problem. For small datasets, we can consider the Multinomial Naive Bayes.

References

[1] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Retrieved from https://www-bcf.usc.edu/~gareth/ISL/ISLR%20First%20Printing.pdf

[2] Loo, M., & Jonge, Edwin de. (2018). Statistical data cleaning with applications in R. Hoboken, NJ: John Wiley & Sons.

[3] Ng, Y. & Jordan, M. On discriminative vs Generative Classifiers: A comparison of Logistic Regression and Naive Bayes. (n.d.) Retrieved from https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

[4] scikit-learn. (2019). Naive Bayes. Retrieved from https://scikit-learn.org/stable/modules/naive_bayes.html

[5] Symantec. (2018). ISTR. Retrieved from https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf

[6] Wikipedia. (2018). Logistic Regression. Retrieved from https://en.wikipedia.org/wiki/Logistic_regression

[7] Wikipedia. (2018). SPAM. Retrieved from https://en.wikipedia.org/wiki/Spam